Comparing Different Functional Allocations in Automated Air Traffic Control Design

Cristian Mattarei, Alessandro Cimatti,
Marco Gario, and Stefano Tonetta
Fondazione Bruno Kessler - Trento, Italy
Email: {mattarei, cimatti, gario, tonettas}@fbk.eu

Kristin Y. Rozier
University of Cincinnati - Ohio, USA
Email: rozierky@uc.edu

***Abstract*—In the early phases of the design of safety-critical systems, we need the ability to analyze the safety of different design solutions, comparing how different functional allocations impact the overall reliability of the system. To achieve this goal, we can apply formal techniques ranging from model checking to model-based fault-tree analysis. Using the results of the verification and safety analysis, we can compare different solutions and provide the domain experts with information on the strengths and weaknesses of each solution.**

In this paper, we consider NASA's early designs and functional allocation hypotheses for the next air traffic control system for the United States. In particular, we consider how the allocation of separation assurance capabilities and the required communication between agents affects the safety of the overall system. Due to the high level of details, we need to abstract the domain while retaining all of the key properties of NASA's designs. We present the modeling approach and verification process that we adopted. Finally, we discuss the results of the analysis when comparing different configurations including both new, self-separating and traditional, ground-separated aircraft.

I. Introduction

By 2025 the airspace will be full [1]; demand for flights will exceed the maximum number of planes that can fly at one time. This problem is not due to a space limitation; there is room for more planes in the air. We will instead exceed the ability of our current system to safely separate commercial aircraft and provide on-the-fly conflict detection and resolution. This is because our current system relies heavily on human air traffic controllers and there is a limit to the number of planes humans can reason about simultaneously. We can solve this problem by adding automation and enabling computers to compute routes and resolutions to maintain safe separation between planes; we already have implementations of optimized algorithms for doing this [2]. But human controllers do much more than 3D geometric reasoning; the entire web of communications between agents in the system, the distributed control structure, and the logical system design all play integral roles in making air traffic control so safe and reliable. If we design a new, more automated system, how do we allocate all of the functions it must perform in a way that upholds at least the current level of safety? This is the *functional allocation* question.

Thanks to the Flight Trajectory Dynamics and Controls Branch of NASA Ames Research Center and NASA's Functional Allocation Project for supporting this work. All models and specifications are available at https://es-static.fbk.eu/projects/nasa-aac/.

The functional allocation question is first and foremost about safety: our goal is to create a partial order on the set of ways to allocate system functions such that system designers can choose a *most safe* configuration and then optimize for secondary goals, such as cost, scheduling, fuel efficiency, ease of use, and environmental impact.

To this purpose, we considered the requirements specification described in the NASA research plan [3] and interacted with NASA engineers to formalize the functions in different allocation configurations, as well as several system requirements to be analyzed. Model checking these properties on different configurations gives us a first means for comparing and ranking the design choices. As a second step, we analyzed functional allocation by extending the model with faults and analyzed the resulting fault trees [4]. Fault trees are commonly used [5] in safety critical contexts, such as aerospace [6], in order to understand which combination of faults can lead to a violation of a safety property. On top of the fault trees, we compute multiple metrics (including probabilistic ones) to compare the different allocation configurations.

We use the NUXMV [7] for model checking and XSAP [8] for fault-tree analysis. Together with the models, the artifacts produced by these tools provide information that goes beyond a mere pass/fail result, providing a rich characterization of the conditions under which properties pass or fail. This enables design choices that minimize the impact of faults on the overall system. To our knowledge, this is the first time that such artifacts have been utilized in the conceptual phase of a real design, when requirements are still blurred and there is no existing concrete design solution to compare with.

Related Work

The complexity of safety-critical systems is continuously increasing. Yet, the current state-of-the-practice is largely characterized by manual approaches, which are error prone, and may ultimately increase the costs of certification. This has motivated, in recent years, a growing interest in techniques for Model-Based Safety Assessment [9]. The perspective of model-based safety assessment is to represent the system by means of a formal model and perform safety analysis, both for the preliminary architecture and at system level, using formal verification techniques. The integration of model-based techniques allows safety analysis to be more tractable in terms of time consumption and costs. Such techniques must be able to verify functional correctness and assess system behavior in the presence of faults [10], [11], [12].

Formal analysis techniques have been applied in the context of NASA's Automated Airspace Concept (AAC) in [13]. That work focuses on analyzing the design proposed in [14] in which the current techniques for Air Traffic Control are extended with automated on-ground support (i.e., TSAFE and Autoresolver). [13] opens the way to the application of symbolic model checking techniques in this context; the analysis is then applied in a probabilistic setting in [15]. In this paper, we start from a more preliminary design proposal (described in NASA research plan [3]) in which we consider the distributed nature of separation assurance in systems were both ground- and self-separated aircraft coexist. To capture the interaction between the different agents, we develop a different modeling abstraction. Moreover, to provide interesting comparative information related to the safety of the designs, we apply safety assessment techniques, such as fault tree analysis.

Other works, e.g., [16], [17] focused on the formal verification of specific functions such as collision avoidance. In this paper, we assume that such functions are correct and we focus on the safety analysis of the overall system.

Another case study using the same tools is presented in [18]. In that case, an avionic wheel braking system is modeled and analyzed according to the standard AIR6110. Different architectures are considered following the process described in the standard. The main differences are that first it is applied on a well-established architecture and not on a design still in the conceptual phase; second, it describes an architecture where the controller is fixed and localized to a specific component instead of relying on a distributed and variable system; finally, the focus on functional allocation addressed in this paper gives more emphasis on some techniques such as the functional analysis of the system reliability in different configurations with respect to the failure probability of a function.

Contributions

The main contribution of this paper is the adaptation of model checking and model-based safety analysis to formally compare different scenarios of an early system design, as required by NASA's functional allocation question. This required a careful definition of the methodology for modeling and fault-tree analysis with the aim of comparing different configurations keeping an abstract view of the single functions. In particular, we handle modeling subtleties in creating a realistic model, such as receptiveness of faults, shadowing, and multiple disjoint communications. Moreover, we verified a list of functional safety features, and computed artifacts describing the reliability of the system with respect to function failures. The outcome of this work allowed us and NASA to reach a better understanding of the design space. Thus, helping NASA shape the work of research groups.

Outline

The rest of the paper is organized as follows: Section II lists the functions and agents that define the functional allocation question and the steps that we followed in the modeling and analysis process; Section III describes the formal model, including the architecture and the abstraction of the real system based on conflict areas and time windows; Section IV describes the properties used to validate the model; Section V describes the formal properties used to characterize the configurations; Section VI explains the use of fault-tree analysis to compare the configurations; Section VII covers modeling subtleties and lessons learned while Section VIII concludes.

II. Functional Allocation for the Automated Air Traffic Control System

A. Problem description

NASA is tasked with designing the next, more automated, air traffic control system for the United States. A major safety goal is to minimize Loss of Separation (LoS), resolve any such situations immediately, and never call upon collision avoidance. LoS occurs when two or more aircraft become too close to each other, i.e., they are below a defined safe distance of 1000 feet vertical and 5 nautical mile horizontal separation. If LoS is not resolved immediately, collision avoidance is necessary. The *functional allocation question* asks which separation assurance (SA) capabilities to require and how to distribute the functions of the design in combination with a subset of these capabilities on top of a set of agents, in order to minimize the number of LoS and the use of collision avoidance techniques [3]. We consider the following agents, functions, and capabilities:

Functions:

- **Strategic Separation** addresses short-term conflicts from 20 minutes in the future down to 3 minutes out from a predicted LoS. Strategic separation is implemented in software and can be running on a central computer on the ground, on-board individual aircraft, or some combination thereof. It uses the trajectories of each known aircraft in the airspace, detecting any conflicts, and outputting resolution maneuvers for any aircraft involved in conflicts.
- **Tactical Separation** addresses near-term conflicts predicted to occur less than 3 minutes in the future. It is also implemented in software running on either a ground computer, an on-board computer, or a combination thereof. Tactical separation must employ a different algorithm from strategic separation because the conflicts it addresses are more imminent and different details must be considered when generating resolution maneuvers.
- **Collision Avoidance** addresses possible collisions less than 30 seconds in the future. Its presence is required by Federal Aviation Administration (FAA) mandate, therefore, TCAS (and in the future ACAS-X), software runs on-board every aircraft, detects possible collisions using a transponder installed in the aircraft, and must operate

totally independently from on-ground systems. A system safety objective is to never trigger collision avoidance.

Agents:

- **Self-Separating Aircraft (SSEP)** carry a separation assurance software on-board.
- **Ground-Separated Aircraft (GSEP)** rely on SA software running on a central on-ground computer transmitting to the aircraft.
- **Air Traffic Control (ATC)** Provides on-ground separation of GSEPs and, when needed, of SSEPs.

Capabilities:

- **ADS-B Out** (Automatic Dependent Surveillance-Broadcast Out) is required on-board all aircraft by FAA mandate by 2020; it broadcasts position information to ADS-B ground stations and other aircraft within transmission range.
- **ADS-B In** is optional by FAA regulations; it receives ADS-B broadcasts from ground stations and other aircraft.

Depending on who is in charge of what, and the available resources, we can describe different designs. Different designs will have different characteristics. Our goal is to provide some qualitative measure of the goodness of each solution along different dimensions. For example, in a scenario in which both GSEP and SSEP aircraft are involved, we might want to know whether a solution in which SSEPs perform both tactical and strategic separation on-board is "better" than a solution in which tactical separation is handled on-ground.

B. *Overview of the process based on formal techniques*

We approach the problem described above using formal methods. We adopted the following four steps process.

1) Modeling (Sec. III): we formalized the system scenarios described in [3]; the informal specification is very abstract and includes only the aspects related to the interaction among the agents; therefore our formalization must choose the right level of abstraction capturing the relevant aspects.
2) Validation (Sec. IV): we performed sanity checks to validate that the formalization of model and properties captures their informal descriptions; in particular, the formal model describing aircraft and controllers are analyzed separately to validate that certain behaviors are allowed.
3) Verification (Sec. V): we formalized the requirements into temporal properties and verified them in the different configurations; some properties must be satisfied by all configurations, but others are used to distinguish and compare different configurations.
4) Safety analysis (Sec. VI): in the previous step, we evaluated the models under nominal conditions, i.e., each component behaves correctly; in the fourth step, each component was extended by adding faulty behaviors, and we evaluated the safety and reliability of the configurations using fault-tree analysis. To compare the different configurations, we analyzed under which failure conditions the system can violate the system requirements.

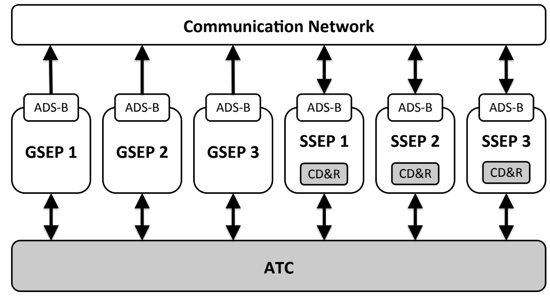

Fig. 1: Scenario instances

III. Formal Modeling for Comparative Analysis

A. *System Architecture*

In this section, we described the model used to analyze and compare different configurations of the functional allocation. The model describes different possible configurations on the number of aircraft. It does not consider the whole airspace, but only the set of aircraft that can be in a conflict on intended trajectories. Both SSEPs and GSEPs aircraft types are taken into account, in addition to ATC and a communication network at airborne level. Figure 1 provides an overview of our model, which allows us to describe a variety of scenarios by enabling or disabling some specific aircraft: only GSEPs or only SSEPs operations (by disabling respectively all SSEPs or all GSEPs), or mixed GSEPs/SSEPs scenario. All SSEP aircraft perform self-separation for the strategic separation with a Conflict Detection and Resolution (CD&R) onboard function, while they rely on the ATC for tactical separation. GSEPs always rely on ground ATC for both tactical and strategic separation. In case an SSEP experiences problems, it is able to ask the ground for strategic separation, thus being treated as a GSEP. The aircraft communicate directly with the ATC while they broadcast messages to other aircraft using the ADS-B. The broadcast is handled by the communication network.

It is important to define the right level of abstraction in order to guarantee that all the relevant aspects are taken into account. In the following sections, we detail what variables define the state of the system, how time passes, and how this influences the change in the state. Note that our analysis focuses on the protocol level and thus, in absence of faults, we assume each component implementation to be correct.

B. *Trajectory Intentions and Conflict Areas*

The basic information that is relevant for our analysis are the trajectories that the aircraft intend to follow, and more specifically if their intentions are in conflict with each other. The actual detail of the trajectories (i.e., the 3D position as a function of time) is not part of our model. In fact, we reason about the system at the architectural level, focusing on the interaction between the components rather than on the precise behavior of the components. We are not interested in which specific trajectory an aircraft should follow to avoid a collision,

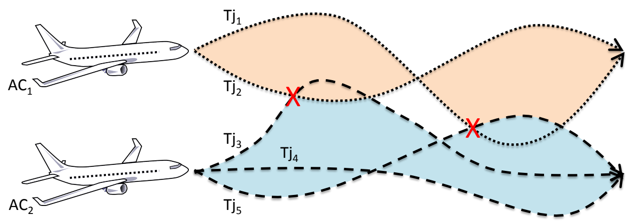

Fig. 2: Conflict Areas abstraction

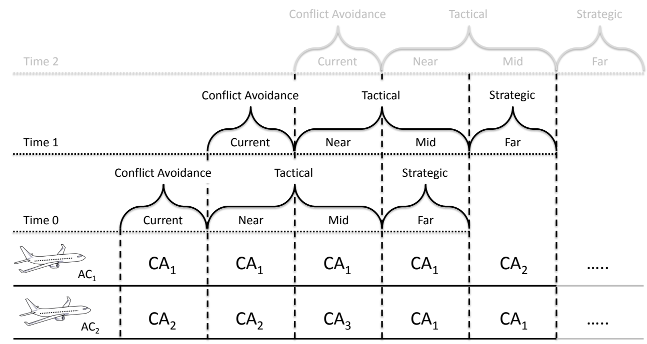

Fig. 3: Near, Mid, Far windows, and their shifting

but only in whether their intentions are in conflict or not. Therefore, we abstract away the detailed trajectory information by introducing *Conflict Areas* (CA). Intuitively, two aircraft are in the same CA, if their trajectories intersect in a given interval of time. In this way, we can abstract the problem of separation into the simpler problem of checking that two aircraft are not in the same conflict area. Figure 2 shows an example when two aircraft have to reach two separate destinations. In this example we consider Tj_1 and Tj_2 for AC_1, and Tj_3, Tj_4, Tj_5 for AC_2. Figure 2 shows that AC_1 and AC_2 are in the same CA if their intended trajectories are respectively Tj_1 and Tj_5, or Tj_2 and Tj_3. In all other cases, they are into different CAs, representing the absence of conflicts. CAs are used throughout our models anytime we talk about aircraft intentions and resolutions sent by controllers.

C. Time windows

Most scenarios in [3] divide the responsibility of the separation-assurance agents based on *time windows*. In particular, we consider four time windows: *Current*, *Near*, *Mid* and *Far*. They represent symbolically consecutive time intervals. Therefore, the trajectory intention of the aircraft define which aircraft are in the same CA in each time window, as defined in the previous section.

The Current window represents the immediate intention of the aircraft, i.e., within 30 seconds. This window is managed by *Conflict Avoidance* algorithms, e.g., TCAS, and is therefore the key to the definition of LoS: two aircraft are currently in LoS if they share the same conflict area in the Current window. The tasks of tactical and strategic separation are then mapped into the Near- and Mid-window (Tactical) and the Far-window (Strategic). If two aircraft share the same conflict area in the same window, we say that we have a *predicted* LoS.

In our model, the intention of aircraft is represented by assigning each airplane with a CA for each window. Figure 3 shows an example with two aircraft. In this example, the aircraft are in different CAs apart from the Far window. So, we have a predicted LoS in that time window.

Intuitively, the windows shift with the passage of time: the old Near information will became the new Current information (Figure 3), while the intention for the other time windows change according to the interaction among the agents. Therefore, if we manage to resolve all predicted LoS, e.g., in the Mid window, we will not have LoS. In order for conflicts to be detected and resolved, we need to take into account the communication between aircraft and the ATC and when it occurs. In the model, passing of time is divided into two main phases that alternate constantly: *communication* and *maneuvering*.

During the maneuvering phase, windows are shifted (Figure 3). During the communication phase, the different agents are able to exchange intentions and resolutions. For example, the aircraft is able to provide its intention to the ATC, and receive a suggestion for a new trajectory. We introduce a bound on the number of communications during this phase, in order to better understand whether multiple iterations between agents can improve the reliability of the system. This interleaving model may seem unintuitive. However, this choice is justified by reality since we can only apply a maneuver after deciding it, and it simplifies the modeling.

D. Scenarios Instantiation

As described in Sec. II, our exemplary scenario allows self-separating aircraft (SSEP), which defines three sub-scenarios: i) non-mixed operations with only GSEPs (current approach); ii) mixed operations with both GSEPs and SSEPs; iii) non-mixed operations with only SSEPs.

We consider a "four aircraft" scenario, where at most four aircraft can be involved in a single conflict at one time. This realistically covers the actual system since conflicts involving more than two aircraft are exceedingly rare [14].

All possible configurations are represented by relying on a single formal model, where each configuration is modeled by enabling or disabling a subset of the components. For instance, considering the model representation shown in Fig. 1, the mixed operation scenario with 2 GSEPs and 1 SSEP is obtained by disabling SSEP 2 and 3.

On top of that, our exemplary scenario describes different possible implementation choices at the communication level that can be enabled (E) or disabled (D):

- GSEP-far: GSEPs send far intentions over ADS-B Out;
- SSEP-far: SSEPs send far intentions to ATC.

Table I shows the size of the different scenarios in term of Boolean variables and AND gates using And-Inverter Graphs (AIG). The last row contains the biggest configuration, which is composed of 353 bits and 4110 AND gates. The first column defines the code used in the rest of the paper to refer to specific scenarios.

Scenario code	Components			# Bool. vars	# AND gates
	GSEPs	SSEPs	ATC		
PA	3	3	✗	283	2226
G	3	0	✓	122	1119
M1	3	1	✓	185	1767
M2	2	2	✓	193	1908
M3	1	3	✓	201	2050
S	0	3	✓	146	1413
ALL	3	3	✓	353	4110

TABLE I: Scenario instances (AIG format)

IV. VALIDATION

The scenarios that are taken into account in this work represent the interaction between a *controller* (the Air Traffic Control and the CD&R on-board of the SSEPs, i.e., the gray components in Fig. 1), and the *controlled system* (the set of aircraft). The objective of this work is to analyze how the Separation Assurance agents control the aircraft. In order to avoid a vacuous verification, we first need to validate separately *controllers* and *system*.

A. Validation Properties Formalization

In order to validate the *system* we identified the following requirements:

VAS-1: It is always possible to reach a LoS.
VAS-2: It is always possible to have no LoS.
VAS-3: It is always possible for an aircraft to maintain the same current intention.
VAS-4: It is always possible for an aircraft to change its intention.

The CTL formalization of the validation requirement VAS-1 is exemplified in (1). More specifically, we want to define that for every state of the system it is always possible to reach a state where Loss of Separation (LoS) holds. The LoS condition applies when at least two aircraft are in the same current conflict area.

$$\begin{aligned} \text{VAS-1} &:= AG(EF(\text{LoS})) \\ \text{LoS} &:= \bigvee_{i \neq j} ac_i.\text{current} = ac_j.\text{current} \end{aligned} \tag{1}$$

The expected behavior of the *controllers* is then defined by the following requirements:

VAC-1: The controller should accept any possible trajectory intent from every aircraft.
VAC-2: The controller should always send a correct resolution.

We validated the model using nuXmv [7], and the results were positive for all 37 properties that formalize the validation requirements.

V. VERIFICATION

The next phase starts with the formalization of a set of properties gathered from the requirements document [3]. We then check whether different configurations satisfy the formal properties. The results of these checks provide us with additional information on the difference between the configurations.

A. Requirements Formalization

We consider the following requirements:

VE-1 It is never possible to reach a Loss of Separation (LoS).
VE-2 It is never possible to have a predicted LoS in the Near-, Mid-, or Far-Window.
VE-3 Every predicted LoS in the Near-, Mid-, or Far-Window is detected by at least one SA agent.
VE-4 Every predicted LoS is detected by at least one SA agent.
VE-5 Resolutions sent by the agent resolve the predicted LoS.
VE-6 Each aircraft must correctly apply the resolution.

The requirement formalization required 93 LTL properties, and their consistency has been validated using the Requirement Analysis Tool [19].

The formal interpretation of the verification property VE-2 is shown in (2), as a composition of the constraints on the near, mid, and far window.

$$\begin{aligned} \text{VE-2} &:= \text{VE-2}_{\text{near}} \wedge \text{VE-2}_{\text{mid}} \wedge \text{VE-2}_{\text{far}} \\ \text{VE-2}_{\{\text{near,mid,far}\}} &:= \bigwedge_{i \neq j} ac_i.\{\text{near,mid,far}\} \neq ac_j.\{\text{near,mid,far}\} \end{aligned} \tag{2}$$

B. Formal Property Verification

All properties are evaluated against all models using nuXmv [7]. The outcome of this evaluation is a table where each cell expresses whether a scenario configuration satisfies a specific property. This allows for a classification of the different possible configurations, and distinguish between the different main aspects that characterize them.

An interesting result is obtained when considering different amount of information that are exchanged between agents. In particular, taking into account the scenario M2, and comparing the configurations E/E and D/D for GSEP-far/SSEP-far implementation choices. In the first case the verification of the requirement VE-2 is fully satisfied, while the latter does not satisfies the sub-requirement over the far window. The motivation of this fact is that each SSEP has the responsibility for the strategic separation, and it requests an ATC support only if it is not able to resolve the conflict. In addition to that, each SSEP computes its intent according to the information provided by other aircraft far intents, but in this case the GSEPs are not providing this information. The result is that each SSEP has not enough information to resolve the conflicts (with GSEPs) in the far window, and the ATC will not provide a backup support because the SSEPs are not requesting the ground support.

TABLE II: Fault descriptions

Comp.	Fault	Description
GSEP/ SSEP	fault_apply_near	Impossibility to apply the suggested trajectory
	fault_apply_mid	
	fault_apply_far	
	fault_comm_atc_par	Communication failure with ATC (partial or total)
	fault_comm_atc_tot	
	fault_comm_adsb	ADS-B In and Out not functional
ATC	fault_near_res	Failure on providing a correct resolution
	fault_mid_res	
	fault_far_res	
SSEP. CD&R	fault_resolve	Failure on generating the resolution
	fault_resolve_detection	Failure on detecting a resolution problem

VI. Safety Analysis

Performing safety analysis of a formal model requires extending the nominal behavior case, i.e., when everything goes as expected, by allowing undesirable behaviors, i.e., failures. The formal model is a representation of a set of requirements, so the occurrence of a fault describes a violation of a system requirement. For instance the constraint describing that each SSEP shall send its trajectory intentions to the ATC holds under nominal conditions, but not in case of a failure (triggered by a specific fault). The safety analysis will then evaluate which faults combinations can lead to an unwanted condition, represented by the negation of a system property, such as Loss of Separation between two aircraft. In safety analysis, such undesired condition is called Top Level Event (TLE). The set of all faults combinations, namely the Cutsets, are usually represented with a tree where leaf nodes are failures, intermediate nodes are AND/OR boolean operators, and the root is the TLE. This artifact is called Fault Tree [4]. A general application of the Fault Tree Analysis considers only the Minimal Cutsets (MCS), and more specifically, a cutset is called minimal if every additional failure will not prevent such undesired behavior.

A. Faults Definition

In this case study, we added several faults for each subcomponent (Table II), in order to check the robustness of each system. For example, we consider different types of communication failure. Aircraft equipped with ADS-B can permanently lose the ability of sending (ADS-B Out) and receiving (ADS-B In) messages (fault_comm_adsb). Similarly, aircraft might lose the ability of communicating with the ATC. In this case, however, we study two different ways in which we can lose communication: permanently (fault_comm_atc_tot) or temporarily (fault_comm_atc_par).

As defined in the requirements documentation, we assume that the components may have the ability to detect the occurrence of some specific faults. For example, a communication link might provide some sort of heartbeat. In these cases, it makes sense to consider some built-in resilience capabilities for the system. For example, if an SSEP realizes that it cannot communicate with the other SSEPs, it will request support from the ATC. The ATC also assumes that if an aircraft does not provide any new intentions due to a failure, then it will follow the most recent ones.

B. Formal Fault Tree Analysis

The analysis of an artifact like a fault tree is important to understand the dependencies between each single component, how they interact, and what is necessary to go wrong in order to not guarantee a necessary behavior of the system.

For instance, we expect that the communication failure of the radio transmission of the ATC can cause a loss of separation between two GSEP aircraft (even if it would be highly improbable). However, we may expect that the loss of communication of a single GSEP cannot cause a LoS, because the ATC would be able to maneuver all the other aircraft in order to avoid him.

For each combination of scenario configuration we computed the associated Fault Tree using xSAP [8], [20]. In the case of property verification, the comparison between two different system configurations is simple, because the property is either satisfied or not. Differently, the fault tree analysis provides very rich artifacts, and there are several techniques that allow us to compare them and define a partial order over the different configurations.

1) Minimal Cutsets Comparison: A common practice in Fault Tree Analysis consists in comparing the size of cutsets of the same cardinality. This approach is based on the intuition that the fewer the single point of failures in the system the higher is the overall reliability. This approach can be extended also to the cutsets of higher cardinality e.g., double failures. This approach provides an intuitive understanding of the relation between different fault trees, however, it is not always precise, since a single failure might be less probable than a double failure.

An example of this analysis is presented in Table III, which compares the results of the FTA on the model instances M1, M2, and M3 (see Table I) when varying the ability to share far intention on the GSEPs (configuration GSEP-far), with the negation of VE-1 as TLE. In this example, the number of single point of failures does not vary for every configurations (i.e., 5), while the number of double failures decreases when the GSEPs share their far intentions with SSEPs aircraft. Important fact, however, is that the number of triple failures increases when GSEP-far is enabled. This behavior in the fault tree analysis results is typical when adding redundant components. In fact the idea behind redundancy is to increase the fault tolerance, and essentially what is a single point of failure becomes a double (or higher) failure.

Further analysis on fault trees can be performed by evaluating the minimal cutsets that are not in common. An example of this analysis can be done by considering the configuration M2 in Table I, and comparing the fault trees obtained with the TLE “there is a LoS between SSEP1 and GSEP1”, when varying GSEP-far.

TABLE III: MCS, ¬VE-1 as TLE, and GSEP-far (E/D)

Card.	**3G-1S (M1)**		**2G-2S (M2)**		**1G-3S (M3)**	
	E	D	E	D	E	D
1	5	5	5	5	5	5
2	12	15	12	16	12	15
3	33	24	35	23	36	27
...	...		...		...	

The results of this evaluation shows that if GSEP-far is disabled then the fault configuration FC $= \{$G1.F_comm_ATC_tot, S1.F_comm_ATC_$tot\}$ can cause the occurrence of the TLE. Differently, when GSEP-far is enabled, FC is no more a necessary condition to reach the TLE because the CD&R on the SSEP is able to react to that situation. In fact, if GSEP-far is enabled then FC requires to be combined respectively with $\{$ATC.$F\_far\_res\}$,$\{$ATC.$F\_future\_res\}$, $\{$G1.$F\_comm\_adsb\}$, and $\{$S1.$cdr.F\_future\_resolve$, S1.$cdr.F\_resolve\_detection\}$ to cause the occurrence of the TLE. Thus, the enabling of GSEP-far turned a minimal cutset of cardinality 2 into 3 cutsets of cardinality 3 and 1 of cardinality 4.

2) Reliability Function Evaluation: A Fault Tree represents all the faults configurations that are necessary to cause the occurrence of the TLE. Assigning a probability of failure to each fault event, and assuming that they are independent, it is then possible to compute the overall probability to reach the undesirable event. More specifically, this approach is based on the generation of the closed form of the reliability function presented in [21]. Such function $P_{\text{TLE}}(f_1, f_2, \ldots, f_n)$ relates the probability of occurrence of the TLE with the failure probability of each fault f_i.

We formally analyze the set of possible AAC designs early in the system design phase, before specific module implementations or probabilities of failures are fully defined. However, we can evaluate how the reliability functions compare to each other by analyzing different possible probability values. For instance, if we take into account the probability of reaching a LoS between two aircraft of the same type (for instance GSEP1/2 and SSEP1/2 in the scenario M2), then we expect that the failure of the ATC will affect more the GSEPs than the SSEPs. This can be assumed considering that SSEP aircraft rely on ATC for strategic separation only as a backup, while they are self-separating otherwise. However, the CD&R on-board SSEPs highly depends on the ADS-B system and its possible failure. Fig. 4 compares the probability of having a LOS (y-axis) between two GSEP (G-G) and two SSEP (S-S), by varying the probability of failure of the ATC (x-axis). We plot the probability functions for different values of the ADS-B failure probability. For the GSEP case, not influenced by ADS-B failures, we obtain a single line, while for the SSEP case we show three different functions. By looking at the intersection points (vertical dashed lines in Fig. 4) this analysis shows us when one solution dominates the others.

The aim of this evaluation is to provide the functions that relate the probability of the TLE occurrence to the probability of failures of each component, and not the actual values of failure probability. In fact, the outcome of the reliability evaluation is a set of functions in Matlab format that can be analyzed using common numerical analysis tools. The remarkable aspect of such type of artifacts is that they do not need to be recomputed when the real component implementation will be defined.

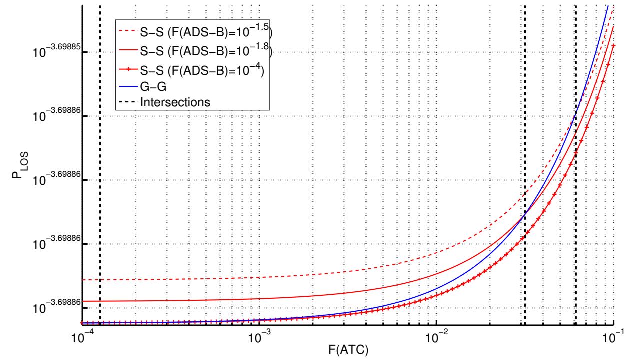

Fig. 4: Reliability comparison between different aircraft types

VII. Lessons Learned

Several subtle technical challenges must be surmounted to complete a realistic, comparative formal analysis on a set of scenarios.

a) Receptiveness of faults: During the fault tree analysis we noticed that some fault configurations were not necessary to cause the reachability of the unwanted condition e.g., LoS. The problem was caused by a chain of relational dependencies through the model, and under some conditions a set of faults $f_1, \ldots, f_n$ imposed f_k to be true. Essentially, we expected both cutsets $cs = \{f_1, \ldots, f_n\}$ and $cs' = \{f_k\}$ to belong to the fault tree FT, thus being minimal cutsets. However, the model implicitly defined the formula $f_1 \wedge \ldots \wedge f_n \rightarrow f_k$, meaning that if cs can cause the TLE then $cs \cup cs' \in FT$. Clearly, cs' is a strict subset of $cs \cup cs'$ which means that even if cs' can cause the occurrence of the TLE then it will not belong to FT, because cs' is not minimal. The solution to this problem was to perform specific receptiveness checks that evaluate if some variables, in this case the fault variables, are always allowed to be assigned to every possible value.

b) Coarse faulty behaviors: Originally, communication faults between ATC and aircraft were not constrained to any specific behavior. This situation caused shadowing in the results of the fault tree analysis. For example, our modeling considers three different time windows: near, mid, and far. For each time window, every aircraft has a trajectory intention and the ATC (or other CD&R components) resolves every conflict in the intentions for every time window. Intuitively, the objective of this design aims to describe a design where a single communication failure in the far window will not cause a Loss of Separation, because it will be possible to resolve the conflict either in the mid or the near window. However, there exists a system execution where the communication failure causes a LoS i.e., when it is total and permanent.

In standard fault tree analysis, each extension of the cutset $\{fault\_communication\}$ will not be considered due to the fact it would be not minimal. The solution to this problem is to refine the model with an additional communication failure, called partial, constrained to a maximum number of occurrences thus providing a more realistic evaluation.

c) Management of multiple communication steps: This work significantly extends the modeling methodology in [13]; for validation, we also modeled the scenario in [13] using the modeling approach described in Sec. III in order to prove that the additional level of detail is able to preserve the previous model's expressiveness. This task was important to discover a weakness in the level of abstraction that defined communication aspects. In fact, in a previous version of the model, the aircraft were allowed to perform a maneuver at every step having a single possible communication step between each maneuver. However, in [13] there is a counterexample where multiple communications directed to an aircraft from different separation assurance agents cause the violation of a property. This system execution, however, is only possible if there are more than one communication steps between each maneuver. Thus, we explicitly allow multiple communication steps.

d) Coarse Top-Level Events: The standard fault tree analysis is strongly characterized by the assumption that each cutset is minimal. This assumption allows us to represent all possible configurations in a compact and intuitive way. However, the choice of a top-level event needs to pay particular attention to this aspect, because the results may not be informative enough. In our analysis we performed the FTA by providing as TLE the negation of a system requirement. For example, our analysis of the requirement that no LoS are allowed between any aircraft provided per se fair results, representing all fault configurations that may cause a LoS. However, the refinement of this top-level event, by expressing each pairwise LoS, provided additional results that were not taken into account previously. More specifically, we expected that the LoS between two aircraft AC_1 and AC_2 can be caused only faults that apply to AC_1, AC_2, and ATC. However, this assumption was not valid in mixed operations when each SSEP aircraft is aware fact that there exists an aircraft that is not able to send the trajectory intentions through ADS-B. In this situation a failure on the aircraft AC_3 can change the behavior of both aircraft AC_1 and AC_2 if they are of SSEP type. As part of this analysis we then decided to enable the possibility to choose if SSEPs aircraft are aware or not of ADS-B failures. The result of the FTA was shadowing important details due to a coarse definition of the TLE, and the solution to this issue was to define more fine-grained TLEs.

VIII. Conclusions and Future Work

This case study provides a first step towards the analysis of the Functional Allocation question. We highlighted a methodology and a series of tools that can be used to analyze and compare different design solutions. In particular, we base the comparison on a set of properties that pass in some configurations and fail in others, on the minimal cut sets obtained with fault-tree analysis, and on the functional dependency of system failure on the failure of single functions. Our approach is expressive, and sufficiently scalable to reason about NASA's full-scale preliminary design space. Important challenges for the future include considering more dimensions of the design space, additional types of faults, and more complex interactions between the agents of the system.

References

[1] MITRE CAASD, "Capacity Needs in the National Airspace System: An Analysis of Airports and Metropolitan Aera Demand and Operational Capacity in the Future," tech. rep., FAA, May 2007.

[2] H. Erzberger, T. A. Lauderdale, and Y.-C. Chu, "Automated conflict resolution, arrival management and weather avoidance for ATM," *Proceedings of the Institution of Mechanical Engineers*, 2011.

[3] T. Lauderdale, T. Lewis, T. Prevot, M. Ballin, A. Aweiss, and N. Guerreiro, "Function allocation for separation assurance: Research plan." NASA HQ Project Overview, Aug. 2014.

[4] W. Vesely, F. Goldberg, N. Roberts, and D. Haasl, "Fault tree handbook," Tech. Rep. NUREG-0492, Systems and Reliability Research Office of Nuclear Regulatory Research U.S., 1981.

[5] "ARP4761 Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment, SAE," Dec. 1996.

[6] W. Vesely, M. Stamatelatos, J. Dugan, J. Fragola, J. Minarick III, and J. Railsback, "Fault Tree Handbook with Aerospace Applications," tech. rep., NASA, 2002.

[7] R. Cavada, A. Cimatti, M. Dorigatti, A. Griggio, A. Mariotti, A. Micheli, S. Mover, M. Roveri, and S. Tonetta, "The nuXmv Symbolic Model Checker," in *CAV*, pp. 334–342, 2014.

[8] B. Bittner, M. Bozzano, R. Cavada, A. Cimatti, M. Gario, A. Griggio, C. Mattarei, A. Micheli, and G. Zampedri, "The xsap safety analysis platform," *CoRR*, vol. abs/1504.07513, 2015.

[9] A. Joshi, M. Whalen, and M. P. Heimdahl, "Modelbased safety analysis: Final report," tech. rep., 2005.

[10] M. Bozzano, A. Villafiorita, O. Åkerlund, P. Bieber, C. Bougnol, *et al.*, "ESACS: an integrated methodology for design and safety analysis of complex systems," *Proc. ESREL 2003*, pp. 237–245, 2003.

[11] O. Åkerlund, P. Bieber, E. Böde, M. Bozzano, M. Bretschneider, *et al.*, "ISAAC, a framework for integrated safety analysis of functional, geometrical and human aspects," *Proc. ERTS*, vol. 2006, 2006.

[12] M. Bozzano and A. Villafiorita, *Design and Safety Assessment of Critical Systems*. CRC Press (Taylor and Francis), an Auerbach Book, 2010.

[13] Y. Zhao and K. Y. Rozier, "Formal specification and verification of a coordination protocol for an automated air traffic control system," *SCP Journal*, vol. 96, pp. 337–353, December 2014.

[14] H. Erzberger and K. Heere, "Algorithm and operational concept for resolving short-range conflicts," *Proc. IMechE G J. Aerosp. Eng.*, vol. 224, no. 2, pp. 225–243, 2010.

[15] Y. Zhao and K. Y. Rozier, "Probabilistic Model Checking for Comparative Analysis of Automated Air Traffic Control Systems," in *Proc. of 33rd ICCAD conference, San Jose, CA, USA, November 3-6, 2014*.

[16] J. Lygeros and N. Lynch, "On the formal verification of the tcas conflict resolution algorithms," in *Decision and Control, 1997., Proceedings of the 36th IEEE Conference on*, vol. 2, pp. 1829–1834, IEEE, 1997.

[17] S. M. Loos, D. W. Renshaw, and A. Platzer, "Formal verification of distributed aircraft controllers," in *Proceedings of HSCC conference 2013, April 8-11, 2013, Philadelphia, PA, USA*, pp. 125–130, 2013.

[18] M. Bozzano, A. Cimatti, A. F. Pires, D. Jones, G. Kimberly, T. Petri, R. Robinson, and S. Tonetta, "Formal design and safety analysis of AIR6110 wheel brake system," in *Proc. of 27th CAV Conference, San Francisco, CA, USA, July 18-24, 2015, Part I*, pp. 518–535.

[19] R. Bloem, R. Cavada, I. Pill, M. Roveri, and A. Tchaltsev, "Rat: A tool for the formal analysis of requirements," in *Computer Aided Verification*, pp. 263–267, Springer, 2007.

[20] M. Bozzano, A. Cimatti, A. Griggio, and C. Mattarei, "Efficient Anytime Techniques for Model-Based Safety Analysis," in *Proc. of 27th CAV Conference, San Francisco, CA, USA, July 18-24, 2015*, pp. 603–621.

[21] M. Bozzano, A. Cimatti, and C. Mattarei, "Automated Analysis of Reliability Architectures," in *18th International Conference on Engineering of Complex Computer Systems (ICECCS)*, pp. 198–207, IEEE, july 2013.